システムソフトウェアと
オペレーティング・システム 92−12
（２００３．２．２８）

PDP11アーキテクチャへのGCC/GAS及びリアルタイムカーネルの移植

飯田 佳洋† 清水 尚彦††

組み込み向け 16 ビットプロセッサとして PDP11 互換プロセッサを開発し、この開発環境としてクロスコンパイラとリアルタイムカーネルの移植を行った。クロスコンパイラ環境 GCC は移植性に優れるがドキュメントの整備が乏しいことからその利用は困難であった。本論文では、PDP11 アーキテクチャへの GCC/GAS の移植内容ならびにリアルタイムカーネルの移植についてを報告する。

Porting Of GCC/GAS And Real-Time Kernel To PDP11 Architecture

Yoshihiro Iida† Naohiko Shimizu††

We developed PDP11 compatible processor as 16-bit embedded processor, and ported cross-compiler and Real-Time kernel as development environment. The cross-compile environment GCC excels in porting but the documents of GCC are incomplete, it is difficult to use. In this paper, we report the porting of GCC/GAS and Real-Time kernel to PDP11 architecture.

1 はじめに

組み込みシステムにおいては、消費電力・コスト・信頼性・市場への安定供給など様々な要求が存在する。製品開発においては、実行性能よりもむしろこれらの要求や開発環境の充実が必要となる場合が多い。

我々は、過去様々なシステムでの動作実績がある PDP11 アーキテクチャを持つプログラマブルチップをターゲットとした組み込み向け PDP11 互換プロセッサを開発した。また、この PDP11 互換プロセッサの開発環境をクロスコンパイラとリアルタイムカーネルの移植を行うことで構築した。

組み込み向けプロセッサとその開発環境はプロセッサベンダのサポートがなくなる場合がある。本報告の PDP11 も 1990 年以降の新機種は発表されていない。我々は、ソフトコアプロセッサとオープンソース開発環境によりベンダから独立した組み込み開発が可能となると考えている。

本論文では、これら開発環境の構築についての報告を行う。

2 PDP11互換プロセッサの開発

PDP11 は、DEC 社が’70〜’90 にかけて市場に供給した 16 ビットミニコンピュータである。コストパフォーマンスが良かった PDP11 コンピュータは UNIX の登場とも相成って広くシステムに導入された。

我々が開発したものは PDP11/40 互換のプロセッサであり、これは、乗除算/拡張算術シフト拡張命令 (EIS)・浮動小数点演算命令 (FPU)・メモリ保護 (MMU) をオプションとして持つ。表 1 は当時の PDP11 と我々が開発した互換プロセッサの主な違いである。これらオプションを持たない互換プロセッサの実装性能を表 2 に示す。図 1 は開発した PDP11 互換プロセッサのブロック図である。

2.1 命令セット

開発した PDP11 互換プロセッサは表 3 に示す命令セットを持つ。

†東海大学工学部通信工学科
Department of Communications Engineering, School of Engineering, Tokai University
††東海大学電子情報学部コミュニケーション工学科
Department of Communications Engineering, School of Information Technology and Electronics, Tokai University

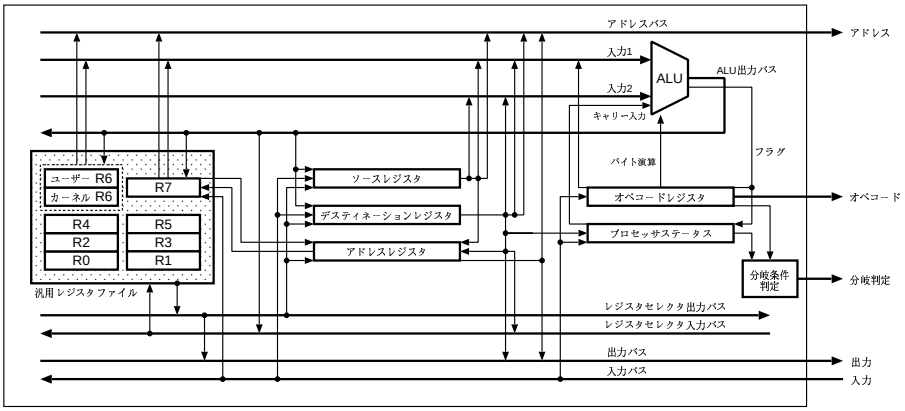

図 1: 開発した PDP11 互換プロセッサのデータパス部

表 3: PDP11 互換プロセッサに実装した命令セット

整数演算命令	**シフト命令**	Negate(byte)	BR. on Greater Than	Return from Interrupt
Add	Arithmetic Shift Right(byte)	Exclusive OR	BR. on Higher	HALT
Add, carry(byte)	Arithmetic Shift Left(byte)		BR. on Lower or Same	WAIT
Sub	Barrel Shift Arith.(Option)	**分岐命令**	BR. on Lower	Condition Code Operaters
Sub, carry(byte)	Barrel Shift 32-bit(Option)	Branch(unconditional)	BR. on Higher or Same	
Multiply(option)	Rotate Right(byte)	BR. on Equal	Jump	**メモリ管理**
Divide(option)	Rotate Left(byte)	BR. not Equal	Jump to Sub Routine	MTPI(Option)
Increment(byte)		BR. on Minus	Return from Subroutine	MFPI(Option)
Decriment(byte)	**データ移動命令**	BR. on Plus	Subtract One and Branch	
Clear(byte)	Move(byte)	BR. on Carry Set		**浮動小数点命令**
Complement(byte)	Swap Bytes	BR. on Carry Clear	**システム命令**	(未実装)
	Sign Extend	BR. on Overflow Set	Set priority level	
整数比較命令		BR. on Overflow Clear	Emulator Traps	
Compare(byte)	**論理演算命令**	BR. on Less Than	Trap	
Test(byte)	Bit Set(byte)	BR. on Greater Than or Equal	Breakpoint Trap	
Bit Test(byte)	Bit Clear(byte)	BR. on Less Than or Equal	I/O Trap	

(捕捉) byte と明記してあるものはワード /バイトの 2 種類のデータを扱うことのできる命令である。

表 1: オリジナルと互換プロセッサの実装面での比較

プロセッサ	オリジナル PDP11/40	PDP11 互換プロセッサ
実装方法	ロジックボード マイクロプログラム方式	FPGA ハード結線
バス	単一バス (Unibus)	複数バス
メモリ	コアメモリ	オンチップメモリ (EABs)
動作周波数	1.1 MHz	50 MHz
割り込み	4 ライン 8 優先度	単一ライン 8 優先度
周辺機器	外付け	オンチップ

表 2: 開発した PDP11 互換プロセッサの FPGA への実装性能

ターゲット	ロジックセル	動作周波数
EPF10K30E	1428 LCs	10 MHz
EP1S80F	1306 LCs	50 MHz

2.2 アドレッシングモード

PDP11 アーキテクチャは表 4 のようにアドレッシングモードの多さが特徴である。メモリ-メモリ演算を持つアドレッシングモードであるため、命令列によるメモリ使用量を削減することができるだけでなく、オンチップメモリを使う場合にはメモリアクセス効率も向上する。

3 PDP11 アーキテクチャへの GNU ツールの移植

PDP11 の開発環境としては Caldera 社より UNIX がオープンソースで提供される。しかし、PDP11 用 UNIX

表 4: アドレッシングモード

アドレッシングモード	ニーモニック	意味
レジスタ	Rn	レジスタ値をそのまま値として
オートインクリメント	(Rn)+	レジスタ値をポインタとして ++
オートデクリメント	-(Rn)	レジスタ値をポインタとして −−
インデックス	X(Rn)	レジスタ値をベースもしくはインデックス値として X と加算
レジスタ間接	(Rn)	レジスタ値をポインタとして
オートインクリメント間接	@(Rn)+	レジスタ値をポインタのポインタとして ++
オートデクリメント間接	@-(Rn)	レジスタ値をポインタのポインタとして −−
インデックス間接	@X(Rn)	計算した実行アドレスをポインタとして
即値	#n	n を 16 ビットの即値として
絶対アドレス	@#A	ラベル A への絶対アドレス指定
PC 相対	A	ラベル A への相対アクセス
PC 相対間接	@A	ラベル A をポインタとして

(注 1) このニーモニック記述は DEC アセンブラ方式である

は言語規格が古く、かつ、高速な汎用マシンによるクロス開発環境の作成が困難であるため GNU ツールの移植を行うことにした。

本稿では PDP11 アーキテクチャへの GNU ツールの移植の経緯をまとめた。

GNU ツールは移植のためのドキュメントの整備がなされておらず断片的なドキュメントと既存アーキテクチャの移植結果を参照しながら移植作業を行う必要がある。PDP11 アーキテクチャに対しては GNU C 3.2 版において断片的な移植が試みられており、コンパイラの生成は出来ないものの移植の試みを軽減することが可能であった。また、GNU binutils も PDP11 アーキテクチャを不完全ながらサポートしているため、これら既存の移植を修正して実際に利用可能にしていく作業を行った。

3.1 GAS と DEC アセンブラ

今回、PDP11 互換プロセッサ用に開発環境を整えるにあたり、GNU binutils 中のアセンブラ (GAS) を利用してクロス開発環境を構築することにする。GAS はほぼ DEC アセンブラとの互換性を有しているが、数値表記その他で互換性を持たない部分がある。そこで、互換性の無い部分に対しては GAS の修正を行うのか GCC からの生成コードを変更するのかを表 5 のように決定した。コンパイラの出力は両アセンブラに互換性を保つようにしたため GCC の出力を DEC アセンブラに掛けることが可能となる。

実アドレッシングモードに含まれないアセンブラ記述用仮想モードのサポートをコンパイラ側で対策したこ

表 5: DEC アセンブラと GAS の非互換性対策

項番	非互換	対策
1	DEC アセンブラは全て 8 進数として扱うが、GAS は C 言語互換の数値表現となる	コンパイラ出力の数値は全て先頭に 0 を付加し DEC/GAS 両互換とする
2	GCC はフレームポインタのデフォルトレジスタ R5 を FP として参照するが GAS では対応していない	GAS を修正すると共に GCC はフレームポインタを FP ではなく R5 として参照するよう修正
3	DEC アセンブラは';' 以降をコメントとして扱うが、GAS は';' を行の区切りと見なす。	コンパイラ出力中のコメントを';' 後、'/*','*/' で囲むことで両アセンブラ互換とした
4	DEC アセンブラは仮想的な記法を有し、アセンブル時に実モードにマッピングする。	仮想モードを生成しないようコンパイラを修正
5	ラベルに対するオフセット、絶対アドレス参照を GAS がサポートしていない	GAS を修正し機能を追加した

表 6: GCC-3.2 の PDP11 ディレクトリ

ファイル名	説明
2bsd.h	2.x BSD 専用ヘッダ類
pdp11-protos.h	pdp11.c で用意する関数群のヘッダファイル
pdp11.c	C で記述する補助関数
pdp11.h	コード生成制御定義類
pdp11.md	マシン記述
t-pdp11	Makefile の固有オプション

とによりコンパイラのコスト計算が変わり出力コードが変化した。GAS を修正して仮想モードをサポートすることで、コンパイラ出力の改良が可能であるが、GAS のアドレッシングモード解析ルーチンに仮想モードを追加すると修正量が多くなるため将来の課題とする。

3.2 GCC 移植

GCC-3.2 の PDP11 ディレクトリには表 6 のファイルが存在する。

今回 BSD アセンブラは用いないため'2bsd.h' 以外のファイルについて修正を行った。GAS と DEC アセンブラの互換性対策以外の修正項目を表 7,8 に上げる。ここに示すように GCC-3.2 の PDP11 ポートは相当不完全であるが、命令数が少ない 16bit CPU への移植例として xstormy とともに貴重な学習教材であると言える。

3.3 C ライブラリ *newlib* の移植

標準 C ライブラリとして組み込み用途に利用可能なコンパクトな *newlib* を移植した。移植にあたっては、既存の他プロセッサの構成を参考にコンフィギュレーションファイルなどの標準的なファイルとディレクトリを用意した。PDP11 用に新たに作成したファイルは実行形式のスタートアップファイルである `crt0.c` ならびにロングジャンプを実現するための `setjmp.S` の二つだけである。組み込み用途を前提に考えているため、システムコールは用意せず基本入出力関数を別途アセンブラ記述で作成し *newlib* から呼び出すことにしている。

`hello.c` をコンパイルするために必要なシステム関数は表 9 の通りである。ライブラリの中で同一コンパイル単位の関数が全てリンクされるため本来必要な関数に比べリンクされる関数の数は多くなる。

表 7: GCC 修正項目 (マシン記述以外)

項番	不具合	対策
1	EIS 未搭載モデルがサポートされない	EIS 未サポートモデル (モデル 10) 用の拡張ライブラリを作成し、モデルオプションで生成コード切り分け
2	浮動小数点サポート必須	soft-float 対応のコンパイルオプション作成し、GNU C 標準小数点パッケージへリンク可能とした
3	浮動小数点ニーモニック不正	訂正
4	スタックポインタ、プログラムカウンタへの間接参照時のアドレス計算不正	アドレス受け入れマクロに特殊ケースとして本モードを追加
5	フレームポインタ未使用時のスタックオフセット不正	オプションを判断しスタックオフセットを自動修正するように変更
6	ソフトフロート時の浮動小数点定数生成不可	ソフトフロート時のみ浮動小数点定数をメモリ定数に変更
7	Makefile 修正	追加ライブラリ定義、マルチライブラリサポート

後述の *proc* の移植において示すように組み込み用途で C ライブラリが不要な場合には *newlib* をリンクするより自前のライブラリをリンクし、`-nostdlib` オプションを用いるべきであろう。

4 リアルタイムカーネル *proc* の移植

オープンソースリアルタイム OS である *proc*[3] は、ほぼ全ての部分が C で記述されるコンパクトなカーネルである。PDP11 アーキテクチャでは最小構成の実行形式は約 900 バイトと小さくなる。PDP11 アーキテクチャへの移植のために 2 つのファイル `pdp11.s`, `stkfpdp11.c` を開発した。これらは、それぞれタスクスイッチングの

表 8: GCC 修正項目 (マシン記述)

項番	不具合	対策
1	浮動小数点レジスタ移動	PDP11 ではメモリと浮動小数点レジスタの間のデータ移動が制限されている。マシン記述全体に修正
2	バイト移動の制限	不要な制限を解除
3	記述誤り	マシン記述中の記述誤り修正
4	バイト論理積生成不可	ビットクリア命令を用いた論理積 RTL 式の生成修正 (expand から insn へ定数反転ロジック移動)
5	XOR 命令不正	オペランド種別を制限して対処
6	シフト命令	未サポートの論理シフト、下位モデルでサポートされない 32 ビットシフト命令、バレルシフト命令に対応するエミュレーション RTL 追加
7	不足 RTL	追加修正

表 9: `hello.c` の実行に必要なシステム関数

項番	関数	説明
1	__main	main 関数初期化処理
2	isatty	ファイルデスクリプタが端末か
3	_read	入力
4	_write	出力
5	_lseek	シーク
6	_close	ファイルクロース
7	_fstat	状態呼び出し
8	_sbrk	データセグメント修正

表 10: *proc* ソースコードの概要

ファイル	行数	バイト数
pdp11.s	233	6425
stkfpdp11.c	98	3967
chan.c	185	5373
chan2.c	125	4398
chan.h	67	2558
proc.c	552	16200
proc.h	332	11071
timer.c	291	8550
timer.h	61	2410
合計	1944	60952

表 11: 使用した EAB のメモリマップ

プロセス	コード領域 (Byte)	データ領域 (Byte)
カーネルライブラリ	2321	118(スタック及び変数)
プロセス 1	137	200(スタック)
プロセス 2	179	200(スタック)
プロセス 3	665	200(スタック)
合計	4020	

コードとスタックフレーム生成のためのコードである。

これらを含め PDP11 アーキテクチャをターゲットとした *proc* を、構築した開発ツールを使ってクロスコンパイルし、FPGA ボードに実装させた。

表 10 は *proc* ソースコードの概要である。これは解説等のコメントアウトも含めた行数を示しているので、実コードはさらに少なく、コンパクトでリアルタイム OS の教育用としても学習に適したソースコード量であるといえる。

開発した PDP11 互換プロセッサと *proc* を用いた組込みシステムをサンプルとして実装した。ALTERA FLEX EPF10K50S チップを使い、FPGA 上の EAB の 4KB をメインメモリとして使用した。表 11 にその 4KB 内のメモリマップを示す。このように、*proc* カーネルを含めた次に示すサンプルソフトウェア及び PDP11 互換プロセッサをプログラム及びワーク領域のメモリを含めて FPGA 上に全て実装した (図 2)。今回、RS-232C 通信による動作情況の出力とスイッチボタン入力を用いたストップウォッチ制御など下記 3 本のプロセスを動作させている。

- 5 個構成の 7 セグメント LED 表示プロセス
- シリーズ LED の周期点灯制御プロセス
- ストップウォッチタイマ計算プロセス

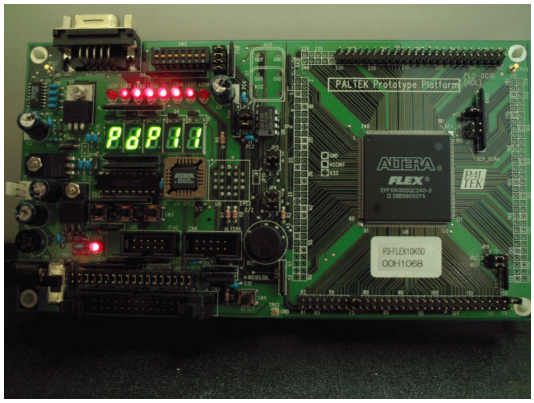

図 2: 実装したボードの概観

これらはプリエンプティブにタスクスイッチングを行い、簡単なリアルタイム性を持った組込みシステムである。

5 まとめ

本論文では、PDP11 互換プロセッサを開発し、それに対する開発環境としての GCC/GAS 及びリアルタイムカーネルの移植について報告を行った。

今後組み込みシステムではより多用なプロセッサコアの需要が高まり、同時にその開発環境の充実も必要となる。GCC 及び *proc* はオープンソースソフトウェアであるため、ベンダのソフトウェア開発方針に左右されないという利点がある。コンパクトなソフトコアプロセッサとオープンソースソフトウェアの組み合わせは組み込みシステム開発に新しい可能性を開くものである。リアルタイムカーネル *proc* は、リアルタイム性を持ちながら非常に小さいオブジェクトコードとなり、ハードウェア資源の制約が厳しい組み込みシステムにもリアルタイム性を持たせることができる。PDP11 互換プロセッサを用いた組み込みシステムの実装を行い、これらがコンパクトに実装できることを確認した。

今後、開発したプロセッサコアを拡張し OS やアーキテクチャ教育にも適用していく予定である。

参考文献

[1] 飯田, 清水, “プログラマブルチップを用いた PDP11 互換プロセッサの開発”, 第 10 回 FPGA/PLD Design conference, 2003

[2] 飯田, 清水, “16bit リアルタイムコンピュータシステム及び開発ツール”, 第 21 回 パルテノン研究会, 2002

[3] *proc* Real-Time Kernel Ver.3.0
http://www.nilsenelectronikk.no/

[4] GNU GCC, GNU binutils, *proc* 移植パッチ
http://shimizu-lab.dt.u-tokai.ac.jp/

[5] “pdp11/40 processor handbook”, DEC, 1972

[6] “pdp11/45 processor handbook”, DEC, 1972

[7] “pdp11 peripherals and interfacing handbook”, DEC, 1972

[8] “Lions’ Commentary on UNIX”, John Lions, ASCII, 1998